Complexity Bounds for the Generalization of Fulton's Intersection Multiplicity Algorithm

Ryan Sandford[*]
Independent Researcher
`rsandfo@uwo.ca`

Abstract. We analyze the Generalization of Fulton's Intersection Multiplicity Algorithm, a symbolic partial algorithm for computing intersection multiplicities via recursive algebraic rewriting. While preliminary benchmarks suggest it may perform well on certain inputs, we show the algorithm exhibits non-elementary worst-case complexity on dense systems and exponential complexity in favorable cases. These results indicate the algorithm's practical utility is confined to systems that are either sparse or of modest size. To address this, we propose an optimization that reduces the depth of recursion across branches of computation, limiting the growth responsible for non-elementary complexity.

1 Introduction

The Generalization of Fulton's Intersection Multiplicity Algorithm (GFIMA) [1, 2, 5] is a symbolic partial algorithm which computes intersection multiplicities by recursively applying algebraic rewrite rules to systems of polynomial equations. GFIMA, along with its extension based on regular chains [3], are implemented in the Regular Chains Library [4] distributed with Maple.

Despite promising preliminary benchmarks [3, 5] against comparable tangent cone and Gröbner basis-based algorithms [7, 8, 6], we show GFIMA exhibits non-elementary worst-case complexity on dense systems, and exponential complexity in favorable cases. These findings suggest that its practical utility is confined to systems that are either sparse or of modest size. We conclude by presenting an optimization that reduces the number of calls to the algorithm's most expensive operation, collapsing the nested compositions which dominate its runtime in general cases.

2 The Generalization of Fulton's Algorithm

2.1 Preliminaries

Let $\mathbb{K}$ be a field and let $\mathbb{A}^n$ denote the affine space of dimension n over $\mathbb{K}$. We define the degree of the zero polynomial to be $-\infty$ with respect to any variable. When $p \in \mathbb{A}^n$ is fixed, we write $f^{(i)} := f(x_1, \ldots, x_i, p_{i+1}, \ldots, p_n)$ to denote the partial evaluation of f at p in the last $n-i$ variables, and $p^{(i)} := (p_1, \ldots, p_i)$ to denote truncation to the first i coordinates. For our analysis, we assume that specializations corresponding to projections $f^{(i)}$ are computed iteratively in reverse variable order, with intermediate results cached and retained unless f is structurally modified. Let $\Phi^{[k]}$ denote the k-fold composition of the function Φ, where $\Phi^{[1]} = \Phi$. The valuation of f at $x_i - p_i$ is defined as the largest m such that $f \equiv 0 \bmod (x_i - p_i)^m$.

[*]This work was conducted independently and does not relate to the author's employment at Amazon.

Definition 1 (Intersection Multiplicity) *The intersection multiplicity of* $f_1, \ldots, f_n \in \mathbb{K}[x_1, \ldots, x_n]$ *at* $p \in \mathbb{A}^n$*, written* $\mathrm{IM}(p; f_1, \ldots, f_n)$ *is defined as the dimension of* $\mathcal{O}_{\mathbb{A}^n,p} / \langle f_1, \ldots, f_n \rangle$ *as a* $\mathbb{K}$ *vector space, where* $\mathcal{O}_{\mathbb{A}^n,p}$ *is the localization of the polynomial ring at* p*.*

Definition 2 (Elimination Degree) *Take* $f \in \mathbb{K}[x_1, \ldots, x_n]$ *and fix* $p \in \mathbb{A}^n$*. We define the elimination degree of* f *at* i*, or equivalently in* x_i*, to be the degree of* $f^{(i)}$ *in* x_i*, denoted by* $\mathrm{elimdeg}(f, i)$*.*

2.2 Complexity Analysis

Algorithm 1 provides a partial algorithm for computing intersection multiplicities by leveraging the generalization of Fulton's properties [1, 2] to rewrite the input system and recursively subdivide the problem. The algorithm has two core components, the Rewrite Loop and the Recursive Split. The Rewrite Loop iterates through every variable, chooses a pivot element with minimal elimination degree in the current variable (say x_j), and uses this pivot element to reduce the elimination degree of up to $n - j$ polynomials. The process of choosing a pivot and reducing the elimination degree of up to $n - j$ polynomials continues until there are $n - j$ polynomials whose elimination degree in x_j is negative. The correctness of this process is ensured by Fulton's rewrite rules, however, when Fulton's rewrite rules cannot be applied, the algorithm terminates and indicates a failure. If the Rewrite Loop is successful, the input system has been rewritten so that it is triangular in the sense that every polynomial f_i corresponds to a unique variable x_{n-i+1}, where the elimination degree of f_i is positive in x_{n-i+1} and negative for all variables greater than x_{n-i+1}. That is, for some fixed solution p and for each index i, x_{n-i+1} divides $f_i^{(n-i+1)}$ with a nonzero quotient. This structure is exploited by the Recursive Split step, which reduces the task to computing an equivalent sum of strictly smaller intersection multiplicities.

2.2.1 Analyzing Expression Swell from the Rewrite Loop

Suppose $f_1, \ldots, f_n$ are dense with maximum degree d_0 and $n > 2$. Observe that for each iteration of the Rewrite Loop (which corresponds to reducing the elimination degree with respect to some x_j), each polynomial can be rewritten at most d_0 times since each rewrite eliminates at least one term with a positive elimination degree in x_j.

Every rewrite performed can increase the maximum degree of the input system. The worst case occurs when every polynomial has d_0 terms with a positive elimination degree in x_j, and when each rewrite eliminates only one term with positive elimination degree in each rewritten polynomial. In this case, a pivot polynomial will be used to reduce the elimination degree of up to $n-j$ polynomials, and one of those rewritten polynomials must then in turn be used to reduce the elimination degree of the pivot polynomial, resulting in a polynomial with degree at most $3d + 1$ where d was the maximum degree before the rewrite cycle. This cycle repeats $d_0 - 1$ times before executing one final rewrite step and proceeding to the next iteration.

Let d_j be the maximum degree at the end of the j-th iteration of the Rewrite Loop. When $j = 1$, the leading coefficient of $f_i^{(j)}$ is in $\mathbb{K}$, hence the degree can increase by at most $d_0 - 1$, thus $d_1 \leq 2d_0 - 1$. When $j > 1$, the leading coefficient of $f_i^{(j)}$ is a polynomial in $j - 1$ variables with a degree which is bounded by the degree of the system. Define $g : d \mapsto 3d + 1$. After $d_0 - 1$ rewrite cycles during the j-th iteration of the Rewrite Loop for $j > 1$, the maximum degree will be at most $g^{[d_0-1]}(d_{j-1}) = 3^{d_0-1} d_{j-1} + \frac{3^{d_0-1}-1}{2}$. Following one final rewrite, the maximum degree at the end of the j-th iteration is bounded by $d_j \leq 3^{d_0-1}(2d_{j-1} + 1) - 1$. Define $h : d \mapsto 3^{d_0-1}(2d + 1) - 1$, a function which models the maximum degree growth after a full iteration of the rewrite loop when

Algorithm 1: The Generalization of Fulton's Intersection Multiplicity Algorithm

```
Function gfima($p; f_1, \dots, f_n$)
    Input: $x_1 \succ \dots \succ x_n$; $p = (p_1, \dots, p_n) \in \mathbb{A}^n$; $f_1, \dots, f_n \in \mathbb{K}[x_1, \dots, x_n]$ where $\mathbf{V}(f_1, \dots, f_n)$ is zero-dimensional at $p$.
    Output: $\mathrm{IM}(p; f_1, \dots, f_n)$ or Fail
    if $f_i(p) \neq 0$ for any $i = 1, \dots, n$ then
        return 0
    if $n = 1$ then
        return $\max\{ m > 0 \mid f_n \equiv 0 \bmod (x_1 - p_1)^m \}$
    for $j = 1, \dots, n-1$ do                                    /* Rewrite Loop */
        while TRUE do
            for $i = 1, \dots, n-j+1$ do
                $r_i \leftarrow \mathrm{elimdeg}(f_i, j)$
            Sort $f_1, \dots, f_{n-j+1}$ so that $r_1 \leq \dots \leq r_{n-j+1}$
            if $r_{n-j} < 0$ then
                break
            $m \leftarrow min(k \mid r_k > 0)$
            for $i = m+1, \dots, n-j+1$ do
                $L_m \leftarrow \mathrm{lc}(f_m^{(j)}; x_j)$
                $L_i \leftarrow \mathrm{lc}(f_i^{(j)}; x_j)$
                $L \leftarrow \mathrm{lcm}(L_i, L_m)$
                if $\frac{L}{L_i}(p) = 0$ then
                    return FAIL
                $f_i \leftarrow \frac{L}{L_i} f_i - (x_j - p_j)^{r_i - r_m} \frac{L}{L_m} f_m$
    for $i = 1, \dots, n$ do                                      /* Recursive Split */
        $q_i \leftarrow \mathrm{quo}\big(f_i^{(n-i+1)}, x_{n-i+1} - p_{n-i+1}; x_{n-i+1}\big)$
    return $\sum_{k=1}^{n} \mathrm{gfima}\big(p^{(k)}; q_{n-k+1}^{(k)}, f_{n-k+2}^{(k)}, \dots, f_n^{(k)}\big)$
```

$j > 1$. Since the loop terminates in $n-1$ iterations, of which only $n-2$ can increase the degree, the maximum degree after completing the Rewrite Loop, is bounded by

$$\begin{aligned} d_{n-1} \leq h^{[n-2]}(d_1) &= 2^{n-3}3^{(n-2)(d_0-1)}(2d_1+1) - \sum_{j=1}^{n-3} 2^{j-1}3^{j(d_0-1)} - 1 \\ &\leq 2^{n-3}3^{(n-2)(d_0-1)}(4d_0-1) - \sum_{j=1}^{n-3} 2^{j-1}3^{j(d_0-1)} - 1. \end{aligned}$$

2.2.2 Worst-Case Complexity of GFIMA

Let m be the intersection multiplicity of $f_1, \dots, f_n$ at point p, and let d_0 be the maximum degree of any f_i. Note that m can be no larger than the Bézout bound d_0^n. Since the sum of the heights of each branch in the recursion tree with positive multiplicity is at most m and each Recursive Split strictly decreases the remaining multiplicity [2], the worst case occurs when $m-1$ recursive calls which return positive multiplicity are made to GFIMA on systems of size n.

Let D_i represent the maximum degree of any polynomial in the input system after i invocations of the GFIMA procedure and suppose $n > 2$. Note that $D_0 = d_0$. Since the i-th invocation of GFIMA executes the Rewrite Loop on a system with maximum degree D_{i-1} we have $D_i \leq 2^{n-3}3^{(n-2)(D_{i-1}-1)}(4D_{i-1}-1) - \sum_{j=1}^{n-3} 2^{j-1}3^{j(D_{i-1}-1)} - 1$ for $i > 0$. Hence, the total complexity of GFIMA is asymptotically dominated by the cost of the polynomial operations performed during the final invocation. The final step of the algorithm computes n specializations in $n-1$ variables on polynomials of degree at most D_{m-1}. After substituting the Bézout bound for m, Theorem 1 gives the final time complexity for GFIMA.

Theorem 1 *If $f_1, \dots, f_n \in \mathbb{K}[x_1, \dots, x_n]$ with maximum degree d, $n > 2$, and $p \in \mathbb{A}^n$, the worst-case time complexity of Algorithm 1 is $O\big(n^2 \left(\Phi^{[d^n]}(d)\right)^n\big)$ where $\Phi : d \mapsto 2^{n-3}3^{(n-2)(d-1)}(4d-1) - \sum_{j=1}^{n-3} 2^{j-1}3^{j(d-1)} - 1$.*

2.2.3 Best-Case Complexity of GFIMA

In the best case, the input system does not require rewriting, reducing the cost of GFIMA to the cost of computing $O(n)$ specializations in $n-1$ variables. This yields a time complexity of $\Omega(n^2d^n)$ if only a single invocation is required, or $\Omega(n^2d^{2n})$ if up to d^n Recursive Split steps are still performed.

Theorem 2 *Let $f_1, \dots, f_n \in \mathbb{K}[x_1, \dots, x_n]$ be polynomials of degree at most d, with $n > 1$ and $p \in \mathbb{A}^n$ such that $f_i(p) = 0$ for all i. Then the best-case time complexity of Algorithm 1 is $\Omega(n^2d^n)$.*

3 Efficient Splitting Optimization

Because the overall cost exhibits non-elementary growth, any optimization that limits the number of Recursive Split steps can significantly improve average-case performance. Theorem 3 achieves this by refining the standard decomposition: rather than decreasing the remaining multiplicity by the sum of sub-multiplicities, it decomposes the remaining multiplicity into a sum of sub-multiplicities weighted by products of valuations. This more aggressive reduction decreases the depth of the recursion tree, hence collapsing the nested compositions responsible for non-elementary growth.

Theorem 3 *Suppose $f_1, \dots, f_n \in \mathbb{K}[x_1, \dots, x_n]$, $p \in \mathbb{A}^n$. If for each i, $\operatorname{elimdeg}(f_{n-i+1}, i) > 0$ and $\operatorname{elimdeg}(f_j, i) < 0$ for any $j < n-i+1$, then $(x_i - p_i)^{v_i}$ divides $f^{(i)}_{n-i+1}$ where v_i is its valuation at $x_i - p_i$. If q_{n-i+1} is the corresponding quotient of each division, we have:*

$$\mathrm{IM}(p; f_1, \dots, f_n) = \sum_{i=1}^{n} \left(\Big(\prod_{j=i+1}^{n} v_j \Big) \cdot \mathrm{IM}(p^{(i)}; q^{(i)}_{n-i+1}, f^{(i)}_{n-i+2}, \dots, f^{(i)}_n) \right).$$

References

[1] W. Fulton, *Algebraic Curves: An Introduction to Algebraic Geometry*, Addison–Wesley, 1989.

[2] M. Moreno Maza and R. Sandford, "Towards Extending Fulton's Algorithm for Computing Intersection Multiplicities Beyond the Bivariate Case," in *CASC 2021*, LNCS 12865, Springer, 2021, pp. 232–251. https://doi.org/10.1007/978-3-030-85165-1_14

[3] R. Sandford, J. Gerhard, and M. Moreno Maza, "Computing Intersection Multiplicities with Regular Chains," *Maple Trans.*, vol. 2, no. 1, 2022. https://doi.org/10.5206/mt.v2i1.14463

[4] F. Lemaire, M. Moreno Maza, and Y. Xie, "The RegularChains Library in Maple," *ACM SIGSAM Bull.*, vol. 39, no. 3, pp. 96–97, 2005. https://doi.org/10.1145/1113439.1113456

[5] R. Sandford, *Towards a Generalization of Fulton's Intersection Multiplicity Algorithm*, M.Sc. thesis, Univ. Western Ontario, 2022. https://ir.lib.uwo.ca/etd/8506

[6] G.-M. Greuel, S. Laplagne, and G. Pfister, "Singular Manual: iMult," 2022. https://www.singular.uni-kl.de/Manual/4-0-3/sing_1277.htm

[7] P. Vrbik, *Computing Intersection Multiplicity via Triangular Decomposition*, Ph.D. thesis, Univ. Western Ontario, 2014.

[8] S. Marcus, M. Moreno Maza, and P. Vrbik, "On Fulton's Algorithm for Computing Intersection Multiplicities," in *CASC 2012*, LNCS 7442, Springer, 2012, pp. 198–211. https://doi.org/10.1007/978-3-642-32973-9_17